Queries

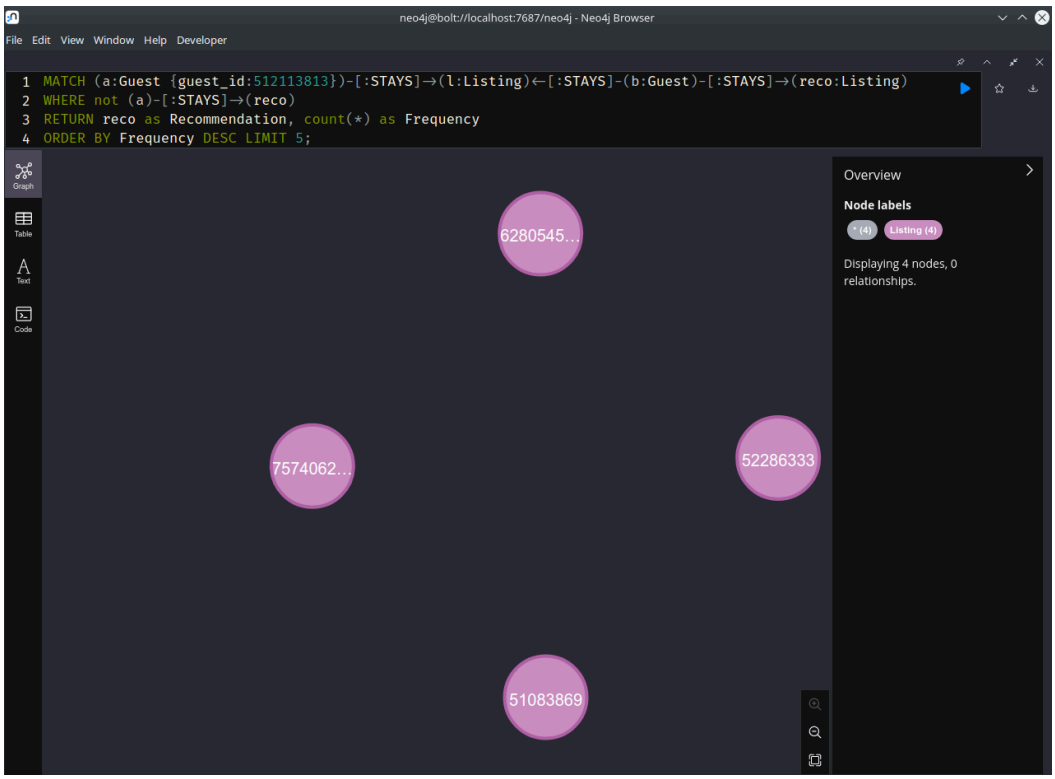
Recommendation System of Listings for a particular Guest

```
MATCH (a:Guest
{guest_id:<guest_id>})-[:STAYS]->(l:Listing)<-[:STAYS]-(b:Guest)-[:STAYS]->(reco:Listing)
WHERE not (a)-[:STAYS]->(reco)
RETURN reco as Recommendation, count(*) as Frequency
ORDER BY Frequency DESC LIMIT 5;
```

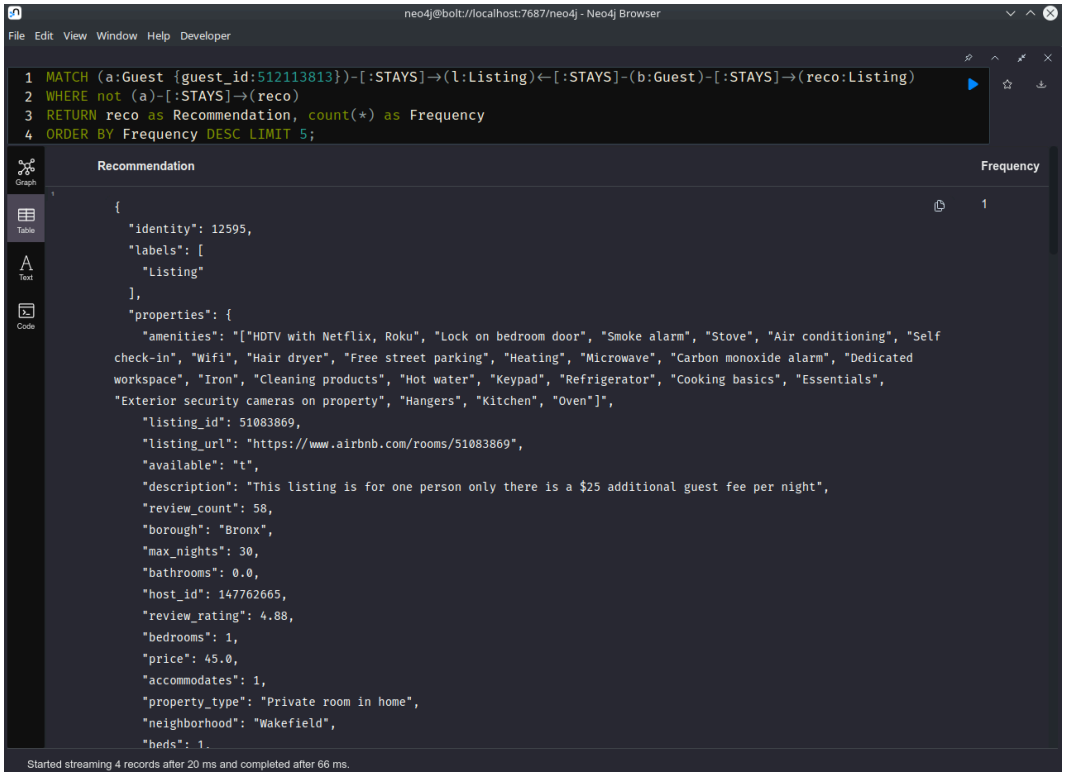
For a given guest, find the top five recommended listings. On Amazon this would be "people who liked this item also liked these other items." In this context, it would be "People who liked the locations you stayed at also liked these other locations."

Finds the listings that the chosen Guest (guest_id) has stayed at, traverses to all other guests who have stayed at the same locations, and collects all locations those other guests have stayed at. Filters away those other locations which the chosen Guest has already stayed at, then orders by the most frequent and limits to the top 5 results.

The below visualization uses guest id 512113813 and returns only 4 nodes, as this guest Jason has only stayed at one listing.



Recommended listings for Jason in graph form.

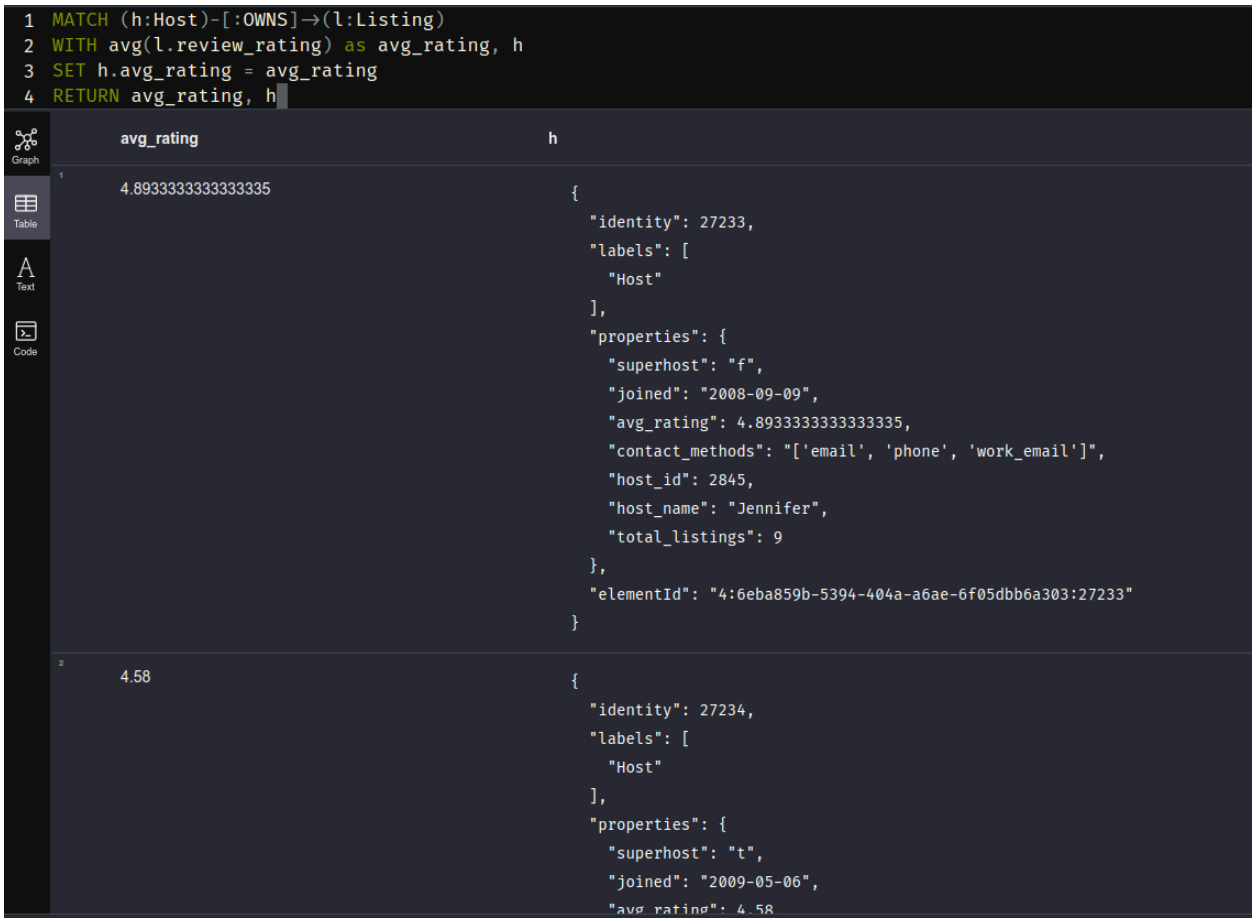


Recommended listings for Jason in table form.

Hosts with average rating above certain threshold with a minimum number of listings

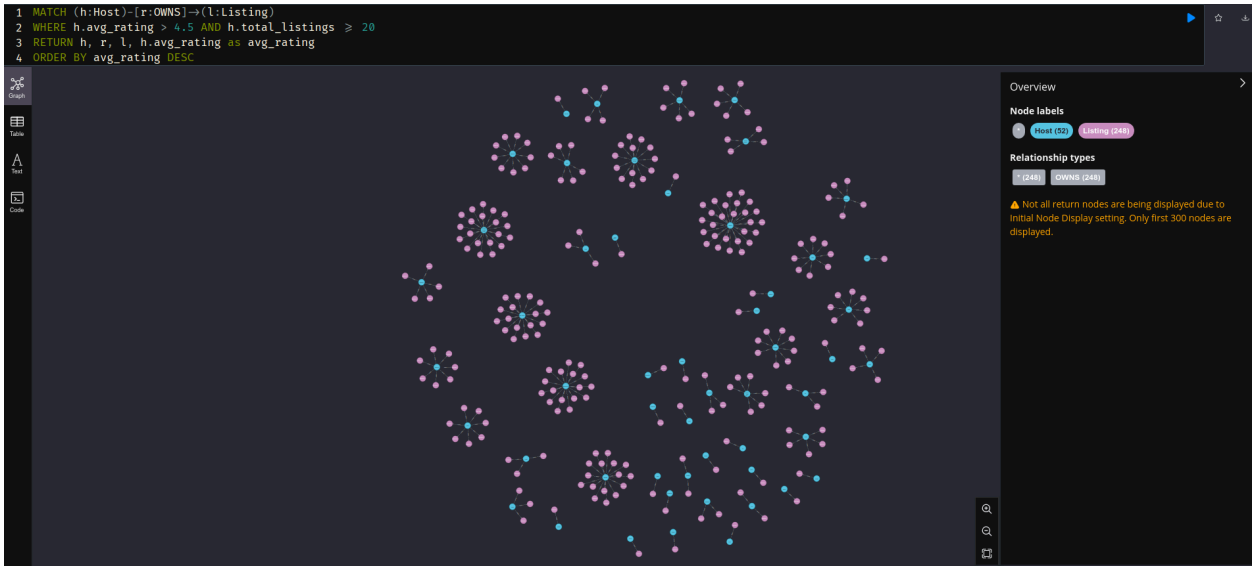
MATCH (h:Host)-[:OWNS]->(l:Listing)
WITH avg(l.review_rating) as avg_rating, h
SET h.avg_rating = avg_rating
RETURN avg_rating, h

This query modifies the database, averaging the ratings for all the listings owned by the host (from the related Listing nodes) and adding it as a new avg_rating property on each Host node.

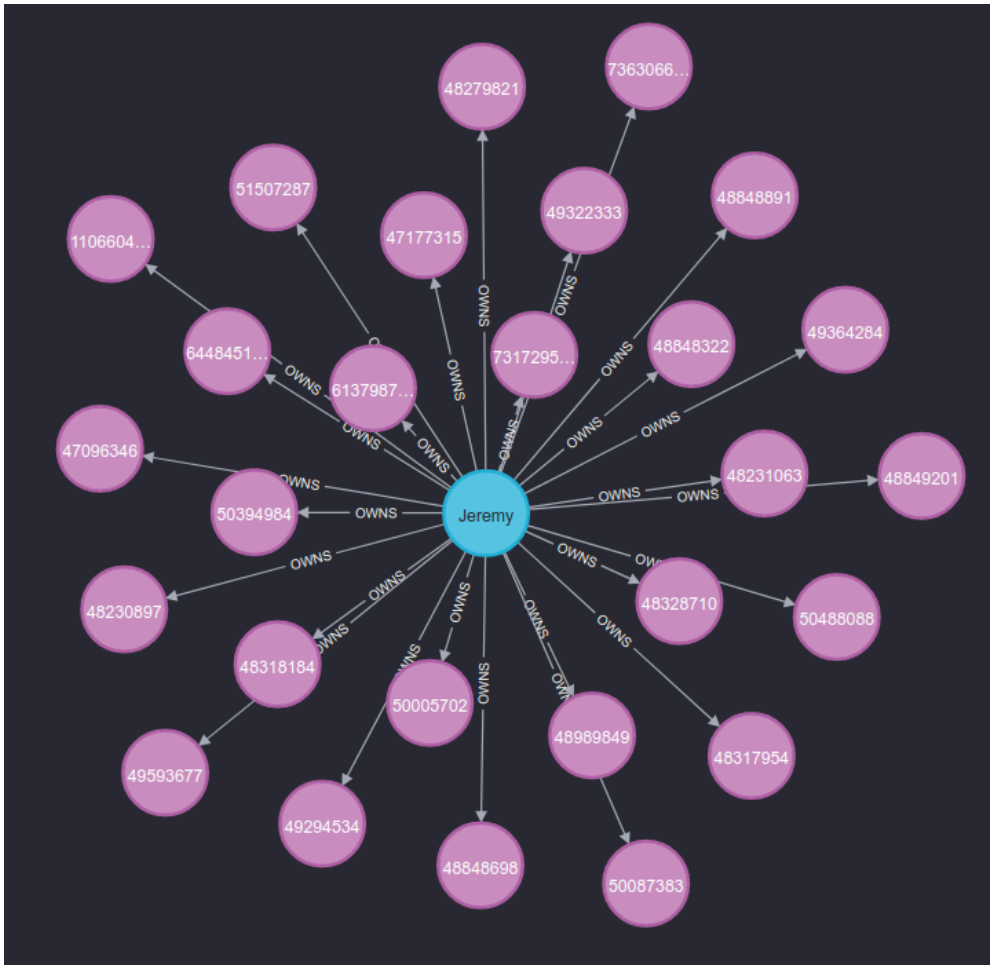


MATCH (h:Host)-[r:OWNS]->(l:Listing)
WHERE h.avg_rating > **<rating_threshold>** AND h.total_listings >= **<min_listings>**
RETURN h, r, l, h.avg_rating as avg_rating
ORDER BY avg_rating DESC

Next, we can use the new property to sort by average rating and collect the Host-Listing relations for display. The query above allows the user to extract all of the hosts with an average rating above a desired threshold (rating_threshold) and all of their listings, as long as they have more than a minimum number of listings (min_listings).



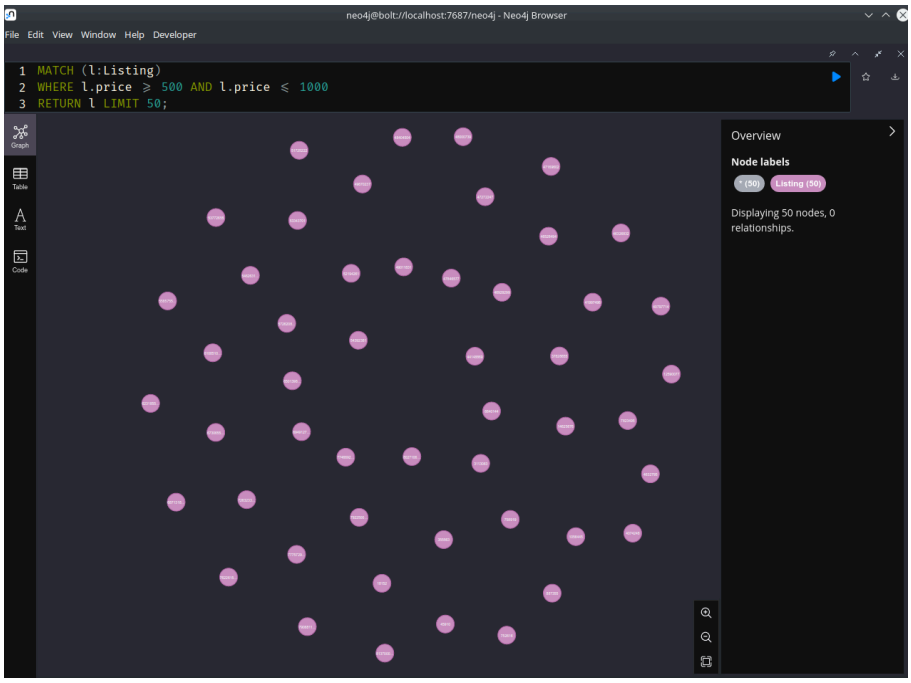
Graph results with avg_rating > 4.5 and at least 20 total listings



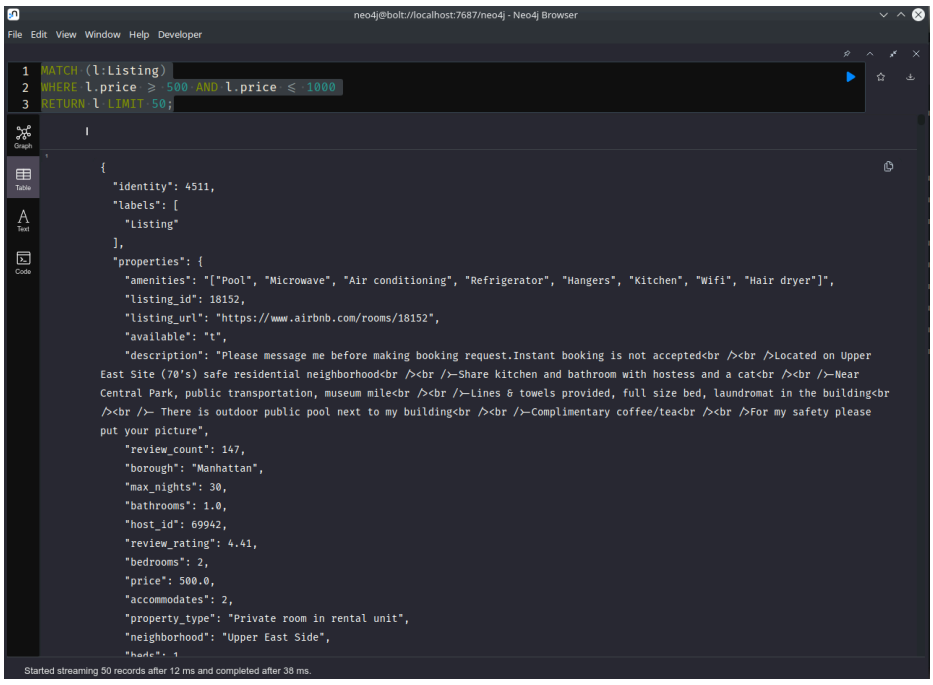
Same results zoomed in on Host 380601266, Jeremy with an average rating of 4.89. Pink nodes are the listings Jeremy owns.

Listings given certain price range

MATCH (l:Listing)
WHERE l.price >= **<min_price>** AND l.price <= **<max_price>**
RETURN l LIMIT 50;



Results for listings between $500 and $1000 in graph form



Results for listings between $500 and $1000 in table form

Neighborhoods with most AirBnBs and those with the most activity (most guest stays)

// Displays neighborhoods, ordered by number of listings descending
MATCH (l:Listing)
RETURN l.neighborhood, count(*) as num_listings
ORDER BY num_listings DESC

```
MATCH (l:Listing)
RETURN l.neighborhood, count(*) as num_listings
ORDER BY num_listings DESC
```

	l.neighborhood	num_listings
1	"Bedford-Stuyvesant"	1698
2	"Midtown"	1457
3	"Hell's Kitchen"	1010
4	"Upper East Side"	1005
5	"Upper West Side"	894
6	"Williamsburg"	893
7	"Harlem"	889
8	"Bushwick"	840
9	"Crown Heights"	733
10	"Chelsea"	520
11	"East Village"	471

Started streaming 222 records after 4 ms and completed after 23 ms.

// Display neighborhoods, ordered by number of stays and number of listings descending
MATCH (l:Listing)
RETURN l.neighborhood, sum(l.review_count) as num_stays, count(*) as num_listings
ORDER BY num_stays DESC, num_listings DESC

```
MATCH (l:Listing)
RETURN l.neighborhood, sum(l.review_count) as num_stays, count(*) as num_listings
ORDER BY num_stays DESC, num_listings DESC
```

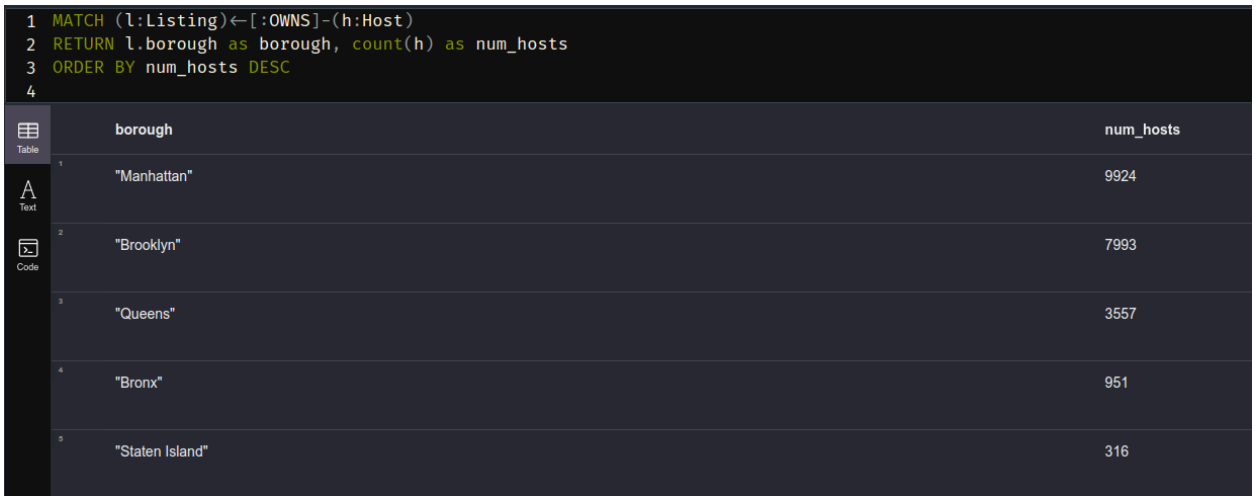
	l.neighborhood	num_stays	num_listings
1	"Bedford-Stuyvesant"	76462	1698
2	"Harlem"	41167	889
3	"Williamsburg"	29933	893
4	"Crown Heights"	28105	733
5	"Midtown"	25368	1457
6	"Hell's Kitchen"	24303	1010
7	"Bushwick"	23488	840
8	"Chelsea"	17831	520
9	"East Village"	17610	471
10	"Upper West Side"	17246	894
11	"Astoria"	15993	264

Started streaming 222 records after 4 ms and completed after 39 ms.

Host Distribution by Neighborhood/Borough

// Neighborhood
MATCH (l:Listing)<-[:OWNS]-(h:Host)
RETURN l.neighborhood as neighborhood, count(h) as num_hosts
ORDER BY num_hosts DESC

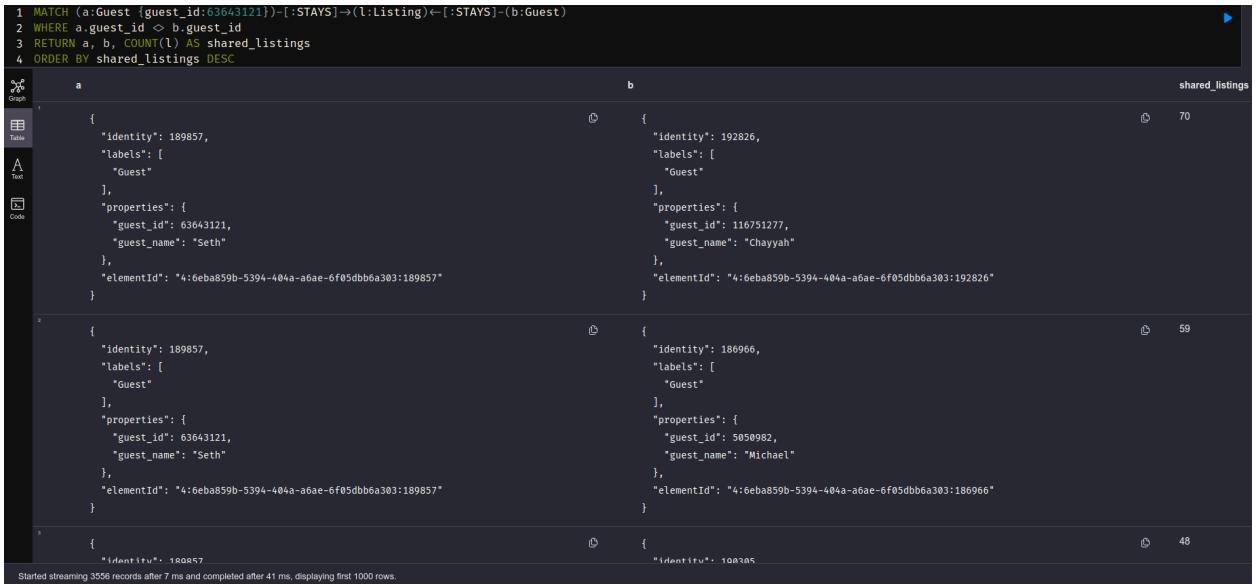
// Borough
MATCH (l:Listing)<-[:OWNS]-(h:Host)
RETURN l.borough as borough, count(h) as num_hosts
ORDER BY num_hosts DESC



Guest-Guest connectivity

MATCH (a:Guest {guest_id:**<chosen_id>**})-[:STAYS]->(l:Listing)<-[:STAYS]-(b:Guest)
WHERE a.guest_id <> b.guest_id
RETURN a, b, COUNT(l) AS shared_listings
ORDER BY shared_listings DESC

Finds the guests that are most connected to the selected guest (chosen_id) by overlap in shared places they have stayed at. This could potentially be used to show guests that are most "similar" to the selected guest. This, combined with further guest metadata, could be used to create guest profile groups for targeted marketing or other uses.



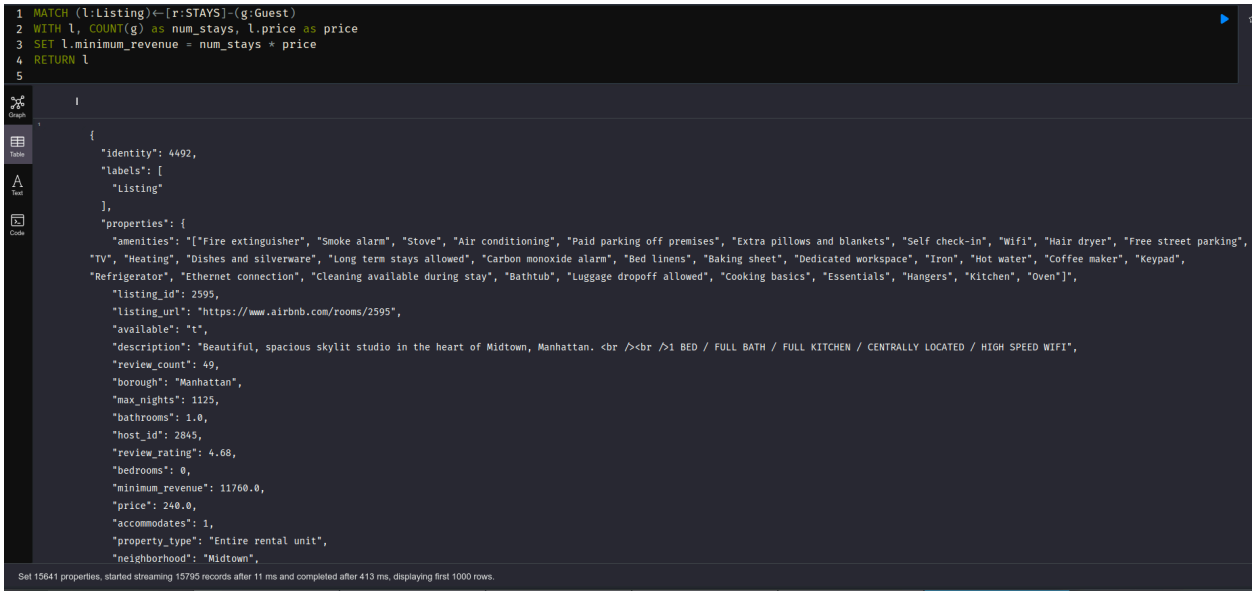
Results for Guest 63643121, Seth.

What AirBnBs are the best investments in terms of revenue?

```
// Adds minimum revenue based on number of stays to nodes with stays
MATCH (l:Listing)<-[r:STAYS]-(g:Guest)
WITH l, COUNT(g) as num_stays, l.price as price
SET l.minimum_revenue = num_stays * price
RETURN l

// Adds minimum revenue of 0 to nodes with no stays
MATCH (l:Listing)
WHERE NOT (l:Listing)<-[:STAYS]-()
SET l.minimum_revenue = 0
RETURN l
```

These operations add a property to each Listing node denoting the minimum revenue for each listing as the product of the number of stays at that listing and the listing price. This is not the full revenue, as the dataset used to generate the database did not include duration of stay for each stay, and each listing price is a daily price. Therefore, the minimum revenue generated is assuming that each stay was exactly one full day in length.



MATCH (l:Listing)
RETURN l.listing_id as id, l.neighborhood as neighborhood, l.borough as borough, l.beds as beds, l.minimum_revenue as minimum_revenue
ORDER BY l.minimum_revenue DESC

Now we can use the new property to perform various queries to explore characteristics of the listings that produce more or less revenue. In the query above, we examine revenue by neighborhood and number of beds in the listings.

```
1 MATCH (l:Listing)
2 RETURN l.listing_id as id, l.neighborhood as neighborhood, l.borough as borough, l.beds as beds, l.minimum_revenue as minimum_revenue
3 ORDER BY l.minimum_revenue DESC
```

Table | Text | Code

	id	neighborhood	borough	beds	minimum_revenue
1	37122502	"East Village"	"Manhattan"	1	610102.0
2	37122162	"Lower East Side"	"Manhattan"	1	402900.0
3	26496505	"Queens Village"	"Queens"	1	399600.0
4	22431770	"Flatiron District"	"Manhattan"	1	365568.0
5	51619634	"Lower East Side"	"Manhattan"	1	342720.0
6	38674685	"Financial District"	"Manhattan"	3	340289.0
7	52795626	"Financial District"	"Manhattan"	2	304380.0
8	691676460109271194	"Chinatown"	"Manhattan"	1	292800.0
9	279857	"Bedford-Stuyvesant"	"Brooklyn"	14	275185.0
10	15324045	"Bushwick"	"Brooklyn"	2	266319.0
11	35545434	"Chelsea"	"Manhattan"	1	246675.0

Started streaming 22741 records after 5 ms and completed after 95 ms, displaying first 1000 rows.

Results of sorting by minimum_revenue generated with some characteristics of listings.

Most Popular Months for Bookings/Stays

MATCH (g:Guest)-[r:STAYS]-(l:Listing)
WITH toInteger(split(r.date,'-')[1]) as month
RETURN month, count(*) as num_stays
ORDER BY num_stays DESC

```
1 MATCH (g:Guest)-[r:STAYS]-(l:Listing)
2 WITH toInteger(split(r.date,'-')[1]) as month
3 RETURN month, count(*) as num_stays
4 ORDER BY num_stays DESC
```

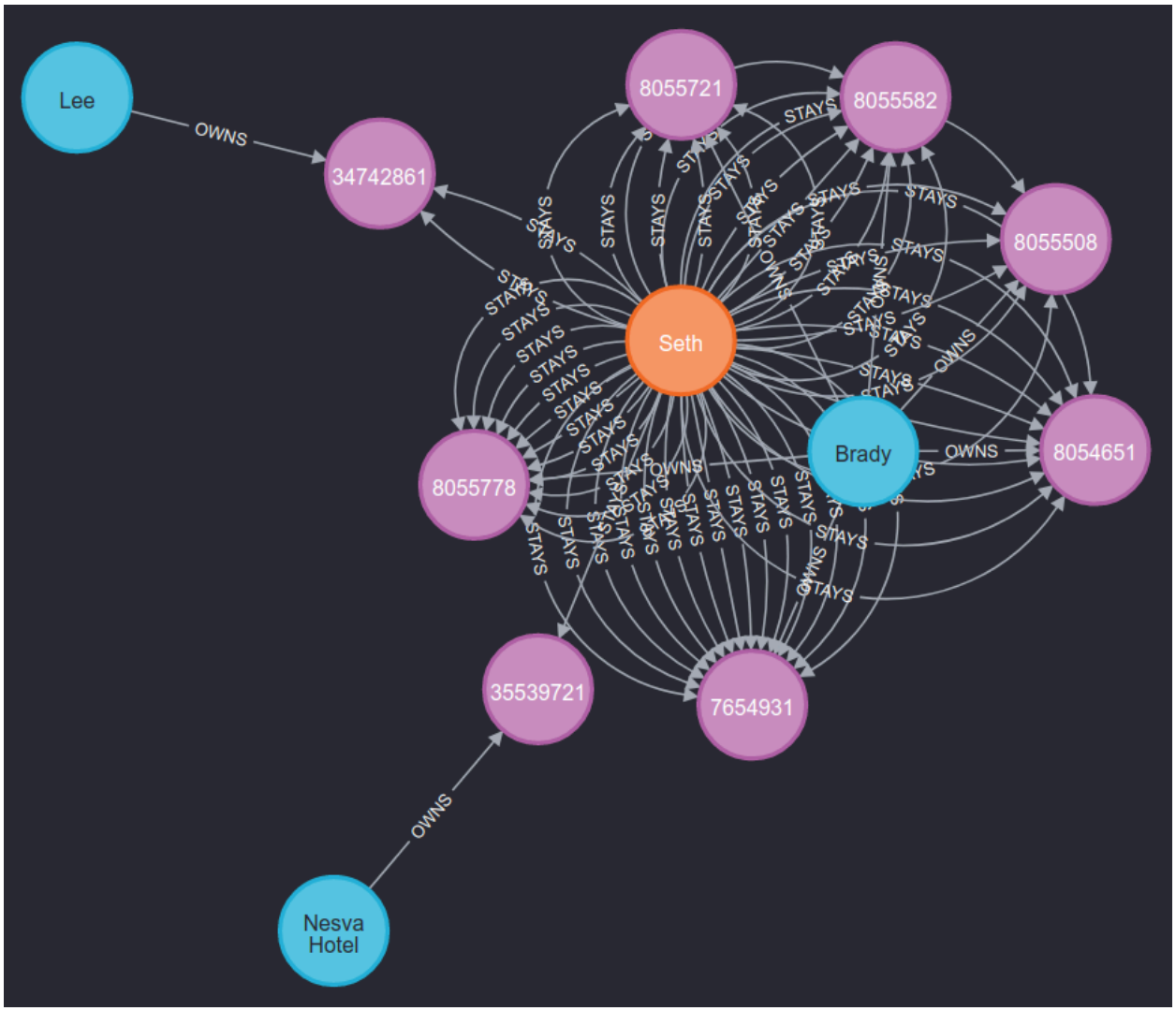
Table | Text | Code

	month	num_stays
1	9	82718
2	10	79417
3	8	76148
4	5	72337
5	7	70481
6	6	69548
7	4	57743
8	11	55025
9	12	54570
10	3	49209

Started streaming 12 records after 6 ms and completed after 421 ms.

Visualizing Guest->Host connections via listings starting from Guest:
MATCH (g:Guest {guest_id:**<chosen_id>**})-[r1:STAYS]->(l:Listing)<-[r2:OWNS]-(h:Host)
RETURN g, r1, l, r2, h

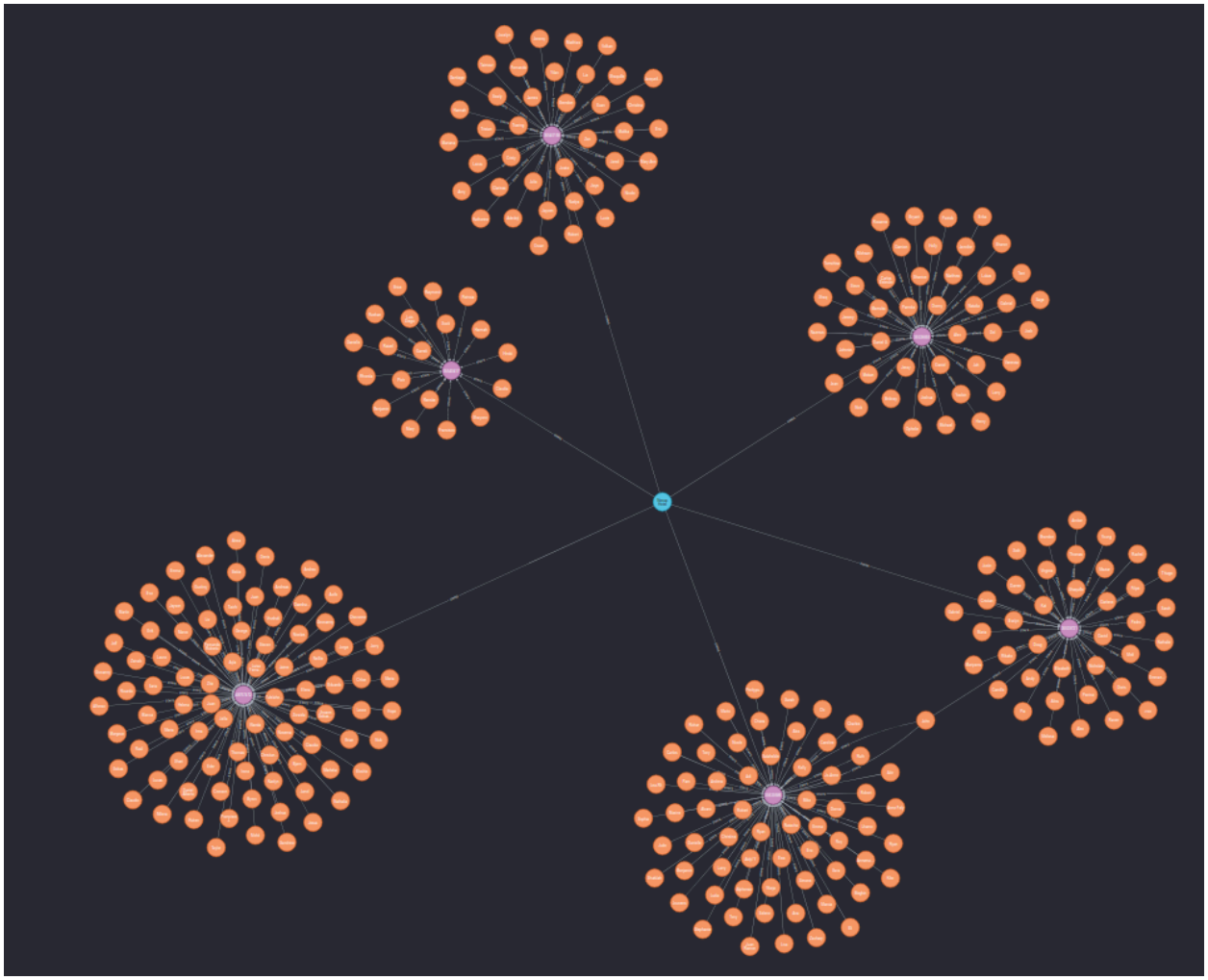
Shows the connections from a selected Guest (chosen_id) to Hosts via all of the Listings that each Guest has stayed at.



Graph results of above query showing Seth's connections using guest_id 63643121. Blue nodes are hosts, pink nodes are listings, and the orange node is the guest, Seth.

Visualizing Host->Guest connections via Listings starting from Host:
MATCH (g:Guest)-[r1:STAYS]->(l:Listing)<-[r2:OWNS]-(h:Host {host_id:**<chosen_id>**})
RETURN g, r1, l, r2, h

Shows the connections from a selected Host(chosen_id) to Guests via all of the Listings that each Host owns.



Graph results of above query showing Nesva Hotel's connections using host_id 263720409. Pink nodes are the listings Nesva Hotel owns, orange nodes are the guests that have stayed at each listing, and the blue node is the host, Nesva Hotel.